UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a master's thesis by

Michael Allen Greminger

nd have found that it is complete and satisfactory in all respects, and that any
nd all revisions required by the final examining committee have been made.

Name of Faculty Adviser

Signature of Faculty Adviser

Date

GRADUATE SCHOOL

# VISION-BASED FORCE MEASUREMENT FOR MICROMANIPULATION AND MICROROBOTICS

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL OF
THE UNIVERSITY OF MINNESOTA
BY

MICHAEL ALLEN GREMINGER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

MAY 2002

# Abstract

When autonomously manipulating objects, force feedback almost always improves the speed and robustness with which the manipulation task is performed. When manipulating objects ranging in size from microns to nanometers, force information is difficult to measure and up to now has been implemented using laser-based optical force measurement techniques or piezoresistive devices. In this paper we demonstrate a method to reliably measure nanonewton scale forces by visually observing the deflections of elastic microstructures. A template matching algorithm is used to estimate the object's deflection to sub-pixel resolution in order to determine the force applied to the object. The template, in addition to containing information about the geometry of the object, contains information about the elastic properties of the object. Applying this technique to an AFM cantilever beam, we are able to measure forces to within +/- 3 nN. Robust estimation techniques are also applied to the template matching algorithm in order to achieve accurate force measurements with up to 30 percent occlusion. Various techniques are used to create the elastic template models including using neural networks to learn from real world data. Vision-based force measurement (VBFM) enables the design of micromanipulators that provide force as well as vision feedback during micromanipulation tasks without the need for incorporating specialized high precision force sensors into the micromanipulation system.

# Table of Contents

# List of Figures

# List of Tables

CHAPTER 1

# Introduction

Due to the limitations of traditional silicon based MEMS manufacturing techniques it is often necessary to use a microassembly step in the manufacturing of functional MEMS devices. This microassembly step allows complex three-dimensional devices to be constructed from two-dimensional MEMS components, and also allows for the manufacture of microdevices using MEMS components produced from incompatible microfabrication processes. Assembly of objects at the microscale requires accurate position as well as force sensing in order to ensure success. Force sensing is important to microassembly and micromanipulation because microparts can easily be damaged by excessive applied loads. Force sensing also provides a sense of “touch” for micromanipulation tasks making it possible to accurately determine contact states throughout a micromanipulation task. Force information is also important when manipulating biological structures such as cells because these objects can be easily injured with improper handling.

To measure forces in the nanonewton range there are two technologies widely used, piezoresistive force sensing and laser-based optical sensing. A piezoresistive force sensor consists of a cantilever beam with an embedded piezoresistive layer that changes resistance as the cantilever deflects [23]. A laser-based optical force sensor makes use of a laser beam that reflects off of a cantilever onto a quadrature photodiode. By measuring the laser beam’s position on this photo diode, it is possible to determine the force applied to the cantilever [16]. Both of these methods are capable of measuring forces down to nanonewton levels or below but require extensive instrumentation to implement. For example, the laser-based aproach requires precise alignment of the laser optics with respect to the cantilever, and the piezoresistive aproach requires a cantilever specially embedded with piezoresistive material and external circuitry to process the output of the sensor. One advantage of vision-based force measurement (VBFM) is that it uses equipment that often already exists in a microassembly or biological manipulation station, including a CCD camera, microscope optics and a computer. Another advantage of this method is that the same technology can be applied to geometries that are more complex than a simple cantilever beam. For example, it can be used with a deformable micro-tweezer to determine how much force is being applied to an object. Implementing a piezoresistive force sensor involves extra processing steps so that piezoresistive material can be incorporated, and using a laser based system would require the incorporation of precisely aligned laser optics into the design of the sensor. VBFM provides a simple method to use an elastic part as a reliable force sensor.

This thesis describes an computer vision approach for measuring force values based on a parameterized template matching technique. This parameterized template encapsulates the part geometry as well as the elastic model of the object that is being used as a force sensor. This elastic model can be as simple as a beam equation, or it can be learned from real world data. Various techniques are used to increase the efficiency of the template matching in order to achieve real-time force sensing performance. Robust statistical techniques are also employed in order to provide robustness to occlusion and noise that may corrupt the source image.

## 1.1 Related Work

The use of elastic models is well established in computer vision. In 1987 Kass et al. [10] proposed a method to track contours in an image using a 2D elastic model called a snake. These snakes had elastic properties and were attracted to edge features within the image. Metaxas [13] used 3D meshes with physics based elastic properties to track both rigid and non-rigid objects. Yuille et al. [28] used a deformable template matching algorithm to track facial features. The difference between these methods and what is proposed in this thesis is that these previous methods used elastic models simply as a tool to locate objects within a scene, while the new approach uses elastic models to actually extract force information from an image.

There also has been work in using elastic models to derive force information, or material property information, from images. Tsap et al. [24] proposed a method to use nonlinear finite element modeling (FEM) to track non-rigid objects in order to detect the differences in elasticity between normal and abnormal skin. They also discussed how their method could be used for force recovery. There has also been work in force measurements at micro and nano-scales using computer vision. Wang et al. [26] used images of microparts to extract CAD models of these parts. They also discuss how they can use the same techniques, along with FEM, to derive the forces that are applied to deformable microparts. Their method is limited by the need to track each FEM mesh point in the image. Danuser et al. [4] proposed the use of statistical techniques along with deformable templates to track very small displacements. They applied their technique to the measurement of strain in a microbar under tension. Dong et al. [6] monitored the tip deflection of an AFM cantilever beam in order to obtain the material properties of multi-walled carbon nanotubes.

## 1.2 Thesis Organization

This thesis is organized as follows. Chapter 2 discusses VBFM applied to a cantilever structure. This chapter also overviews the parameterized template matching algorithm that is used to perform the visual force measurement. Chapter 3 extends the concept of VBFM to a microtweezer in order to demonstrate the adaptability of this approach to other devices and geometries. Next, Chapter 4 discusses the optimizations that were used in order to achieve real-time force sensing. The use of robust estimation techniques to make the algorithm more robust to occlusion and noise are discussed in Chapter 5. Chapter 6 discusses how neural networks can be used to extend the capabilities of VBFM so that it can be used with objects where the elastic properties are not known in advance. The concluding discussion is given in Chapter 7.

CHAPTER 2

# VBFM Applied to a Cantilever Beam

## 2.1 System Overview

A diagram of the force sensing system as it is applied to a cantilever beam is shown in Figure 1. The cantilever beam is an AFM probe tip 450 μm long with a spring constant of approximately 0.1 N/m. A known displacement is applied to the cantilever beam using a 3 DOF piezo-actuated nanopositioner manufactured by Queensgate with sub-nanometer positioning resolution. A microscope mounted with a CCD camera is used to obtain an image of the cantilever. The view from the camera is shown in the right half of Figure 1. A video capture card then digitizes this image. A Canny edge operator is applied to the image to get a binary edge image [3]. Next a template matching algorithm is used that computes a virtual force that is applied to the cantilever. The virtual force value returned by the template matching algorithm, once calibrated, gives the force that is applied to the cantilever. Figure 2 shows a flow diagram of the VBFM aproach applied to a cantilever beam.

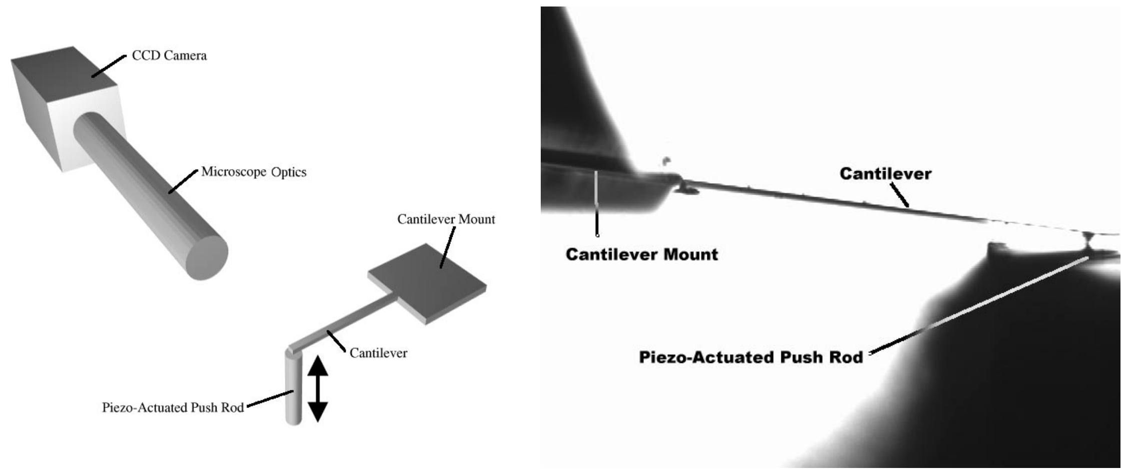

**Figure 1. Force sensing system configuration on the left and actual view from the camera on the right.**

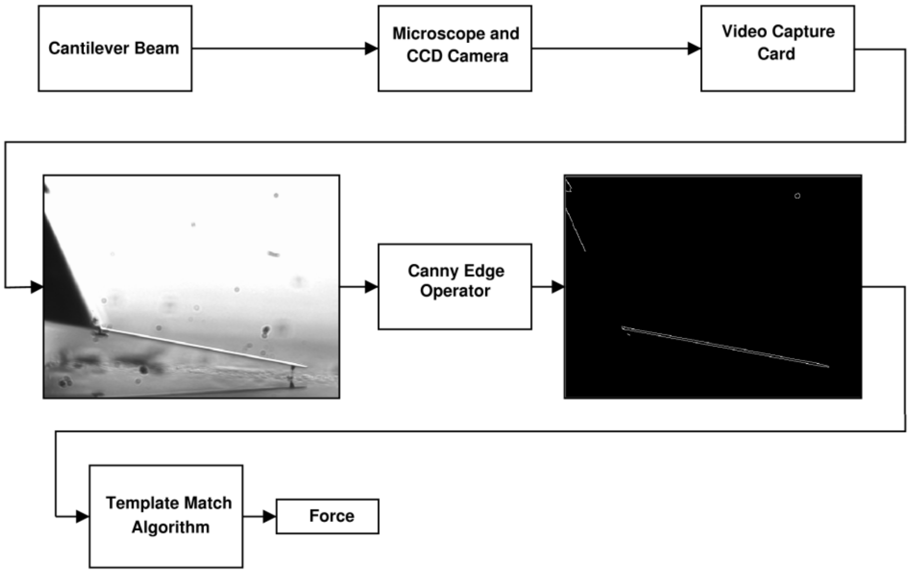

**Figure 2. Flow diagram of force measuring system.**

## 2.2 Template Matching Algorithm

Template matching is a method used to locate a known pattern within an image. A representation of the pattern that is to be matched, called the template, is compared with an image to determine where that pattern occurs within the image.

### 2.2.1 Traditional Template Matching Algorithm

In the traditional template matching algorithm [18], a template is scanned over the entire image. At each location $(m, n)$ the difference between the template and the image is evaluated by an error function $E(m, n)$. This function is defined as

$$E(m, n) = \sum_j \sum_k [I(j, k) - T(j - m, k - n)]^2 \qquad (1)$$

where $I(j, k)$ is the intensity of the image at the point $(j, k)$, and $T(j, k)$ is the intensity of the template. The values $j$ and $k$ iterate over the region where the template overlaps the image. The point on the image with the lowest value of the error function indicates the location of the pattern within the image. To insure that the pattern is not found when it is not in the image, a threshold value $E_{max}$ for the error function must be set. Therefore, the pattern is found at the point $(m, n)$ if the error function is a minimum at this point and has a value less than $E_{max}$.

There are limitations to this template matching algorithm, the biggest being that many templates need to be used if the pattern is not constrained to a particular orientation or size within the image. Another limitation is that the location of the pattern within the image can only be found to a resolution of one pixel. This is true because the evaluation of the error function only makes sense if the pixels of the template and the image overlap directly. The most constraining limitation of this algorithm for visual force detection is its inability to find objects that change shape.

### 2.2.2 Parameterized Template Matching Algorithm

As mentioned previously, one problem with the traditional template matching algorithm is that a feature can only be located to the nearest pixel. In order to obtain sub-pixel matching resolution, it is necessary to compose an error function that does not require the template pixels and the image pixels to line up precisely, as does the error function used for the traditional template matching algorithm in (1).

In order to compose a new error function, it is necessary to represent the template and the image each as a list of vertices rather than a 2-D array of pixels. To do this, the template and the image are assumed to be binary, with each pixel that is on representing an edge and each pixel that is off representing the background. Note that the traditional template matching algorithm allowed the template and image to be binary, grey scale or even color. Since the template is a binary image, we can represent it by a list of vertices with each vertex corresponding to the coordinates of an edge pixel in the template. The image similarly will be represented by a list of vertices with each vertex corresponding to the coordinates of an edge pixel in the image. The template vertices are defined with respect to a local coordinate system, the $\xi - \eta$ coordinate system, and the image vertices are defined with respect to the original image's coordinate system, the $X - Y$ coordinate system (see Figure 3). In order to be able to compare the template vertices to the image vertices it is necessary

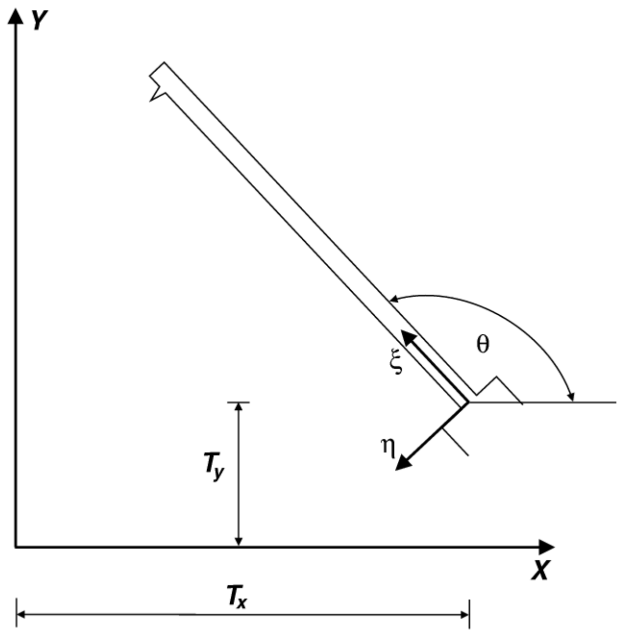

**Figure 3. Beam template shown with respect to image coordinate system.**

to map the template vertices from the $\xi - \eta$ coordinate system to the $X - Y$ coordinate system. This can be done using the homogeneous transformation given by

$$\begin{bmatrix} x_T' \\ y_T' \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} x_T \\ y_T \\ 1 \end{bmatrix} \tag{2}$$

where $(x_T, y_T)$ is a template vertex coordinate with respect to the $\xi - \eta$ coordinate system, $(x_T', y_T')$ is the template vertex coordinate with respect to the $X - Y$ coordinate system, and $\boldsymbol{A}$ is a homogeneous transformation matrix. The matrix $\boldsymbol{A}$ is composed of a scale, rotation and translation. $\boldsymbol{A}$ can be written as

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & T_X \\ 0 & 1 & T_Y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S & 0 & 0 \\ 0 & S & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S\cos\theta & -S\sin\theta & T_X \\ S\sin\theta & S\cos\theta & Ty \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

where $S$ is the scale factor of the template about its origin, $\theta$ is the rotation of the template about its origin, $T_X$ and $T_Y$ are the $x$ and $y$ components of the translation of the origin of the template coordinate system with respect to the image coordinate system. One can see that $\boldsymbol{A}$ is a function of $S$, $\theta$, $T_X$, and $T_Y$.

The new error function is therefore given by

$$E(S, \theta, T_X, T_Y) = \sum_{j=1}^{N} [(x_{Tj}' - x_{Ij})^2 + (y_{Tj}' - y_{Ij})^2] \tag{4}$$

where $(x_{Tj}', y_{Tj}')$ are the coordinates the $j$ th edge pixel of the template transformed by (2); $(x_{Ij}, y_{Ij})$ are the coordinates of the edge pixel in the image that is closest to the point $(x_{Tj}', y_{Tj}')$; and $N$ is the number of edge pixels in the template. This error function sums the distance squared between each of the template edge pixels and the closest edge pixel on the image. Since the coordinates $(x_{Tj}', y_{Tj}')$ are related to the template coordinates $(x_{Tj}, y_{Tj})$ by the homogeneous transformation matrix $A$, $E$ is a function of $S$, $\theta$, $T_X$, and $T_Y$. By minimizing (4), the values of $S$, $\theta$, $T_X$, and $T_Y$ that best match the image in a least squares sense can be determined.

This formulation of the error function overcomes many of the limitations of the of the traditional template matching algorithm. Since $T_X$ and $T_Y$ are not constrained to be integer values, the pattern can be located to resolutions smaller than one pixel. Also, since the error function depends on scale and rotation, as well as $x$ and $y$ translation, only one template needs to be used even if the orientation and size of pattern to be located are not known in advance. This algorithm does not allow for objects that change shape, which will be discussed in Section 2.3.

### 2.2.3 Minimizing the Parameterized Template Matching Energy Function

The error function given by (4) is minimized by a first-order multi variable minimization technique that uses the gradient of the error function in the minimization process. The first-order method used is called the Broydon-Fletcher-Goldfarb-Shanno (BFGS) method [25]. The choice of the minimization algorithm is very important to the overall performance of the template matching routine and it will be discussed in more detail when discussing optimizations in Chapter 4.

## 2.3 Incorporating Force into the Template Matching Algorithm

The template matching algorithm is only able to locate rigid objects within the image. It is not possible to determine the force applied to the cantilever beam with such a template. In order to overcome this limitation, it is necessary to incorporate the elastic properties of the object into the template. This is done by deforming the original template by the appropriate elastic model.

For the cantilever beam, the Bernoulli-Euler law is assumed to apply [21]. The Bernoulli-Euler law can be written as

$$R = \frac{EI}{M} \tag{5}$$

where $E$ is the modulus of elasticity of the beam, $I$ is the moment of inertia of the beam, $M$ is the moment being applied to the beam and $R$ is the radius of curvature of the neutral axis. The Bernoulli-Euler law says the radius of curvature of the neutral axis of the beam is proportional to the bending moment applied to that part of the beam. The Bernoulli-Euler law can be used to show that the deflection along the length of the cantilever beam shown in Figure 4 is

$$y = \frac{Fx^2}{6EI}(3L - x) \tag{6}$$

where $F$ is the force being applied to the beam, and $L$ is the length of the beam. The assumptions of the Bernoulli-Euler law are the following: 1.) The material of the beam is linear elastic, isotropic and homogeneous, 2.) The strains and displacements are infinitesimal, and 3.) There is no shear deformation (the beam is in pure bending). Assumptions two and three can be used with long and slender beams such as the AFM probe tip used in this force sensor. Assumption one is assumed for the silicon from which the beam was etched.

Next, it is necessary to apply this elastic model to the beam template. It can be seen from (6) that the $x$ values for each of the template points will remain unchanged. It should also be noted that if a template point's $x$ value is less than zero, then the point will not be

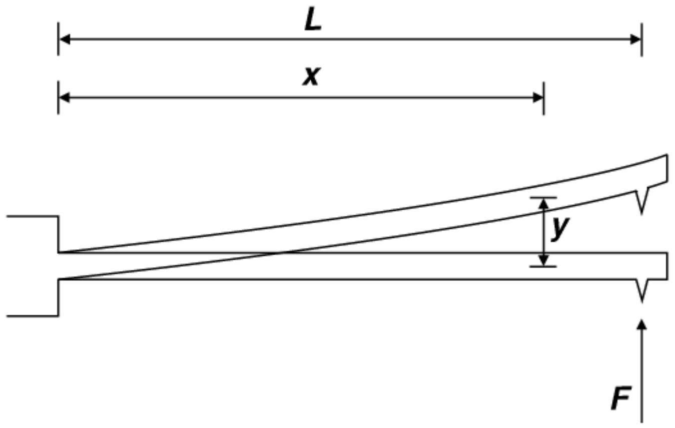

**Figure 4. Beam deflection model.**

transformed. If a template point's $x$ value is between zero and $L$, than the point will be translated in the $y$ direction by the amount given by (6). For points with an $x$ value greater than $L$, they will be translated in the $y$ direction but the equation will be different from (6) because the radius of curvature of the beam becomes infinite for $x > L$. Once all of the template points are translated appropriately, then (2) can be applied to the new template points to transform them to the image coordinate system. This is shown by the following equations:

$$\begin{bmatrix} x_T' \\ y_T' \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} x_T \\ y_T \\ 1 \end{bmatrix} \text{ for } (x_T < 0) \tag{7}$$

$$\begin{bmatrix} x_T' \\ y_T' \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} x_T \\ \frac{Fx_T^2}{6EI}(3L - x_T) + y_T \\ 1 \end{bmatrix} \text{ for } (0 \le x_T < L) \tag{8}$$

$$\begin{bmatrix} x_T' \\ y_T' \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} x_T \\ \frac{FL^2}{6EI}(3x_T - L) + y_T \\ 1 \end{bmatrix} \text{ for } (x_T \ge L) \tag{9}$$

Figure 5 shows an undeformed beam template acquired from the image of a beam on the left and the same template with a template that was deformed using (7) through (9) on the right. When the error function (4) is used with a template transformed by (7) through (9), the error function becomes a function of one additional variable, the force $F$ applied to the template. This is shown by the following error function:

$$E(S, \theta, T_X, T_Y, F) = \sum_{j=1}^{N} [(x_{Tj}' - x_{Ij})^2 + (y_{Tj}' - y_{Ij})^2] - \tag{10}$$

When this error function is minimized, in addition to giving the position of the beam within the image, the force that is being applied to the beam can be obtained. Figure 6 shows a deformable template matched to an image of a cantilever beam that is being deformed.

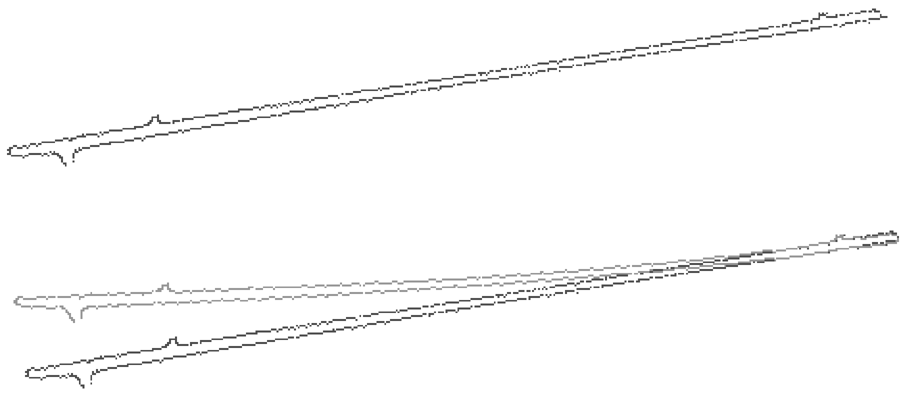

**Figure 5. Original beam template on the top and deflected beam template with original beam template on the bottom.**

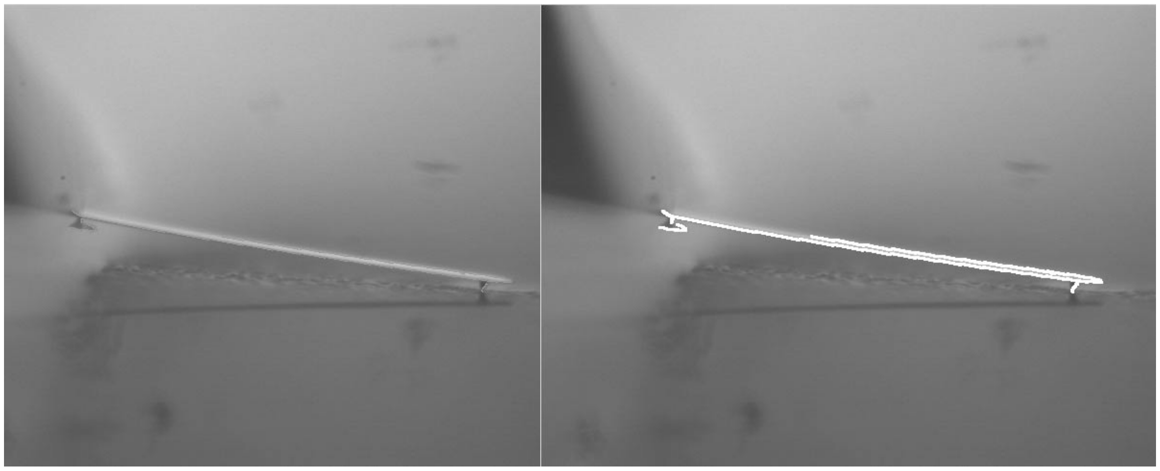

**Figure 6. Deflected cantilever on left and template matched to deflected cantilever on right.**

## 2.4 System Calibration

It can be seen from (6) that the relationship between the deflection of the tip ($x = L$) of the cantilever and the force applied to the cantilever is:

$$y = \left(\frac{L^3}{3EI}\right)F \tag{11}$$

This shows that the displacement at the tip of the beam is directly proportional to the force applied to the beam. Because of this linear relationship, the force sensor can be calibrated by applying a known displacement to the beam and then scaling the virtual force obtained from the template matching algorithm so that it represents the displacement of the tip of the cantilever. Once the system is calibrated to measure the beam tip deflection accurately, (11) can be used to determine the force that was applied to the beam.

The above discussion shows that the accuracy of the force reading obtained by the force sensor is not only limited by the accuracy of the vision algorithm, but is also limited by the accuracy of the beam parameters $L$, $E$, and $I$ (where $I$ depends on the height and width of the cross section of the beam, $h$ and $b$ respectively). Solving (11) for force and substituting $(bh^3)/12$ for the moment of inertia $I$, the following equation is obtained for force in terms of the deflection of the end of the beam:

$$F = \frac{Ebh^3}{4L^3}y \tag{12}$$

All of the variables in the above equation have some uncertainty. The uncertainty, $\Delta F$, in the force value depends on the uncertainties of all of the variables in (12). The uncertainty in $F$ can be calculated from the uncertainties of the variables in (12) using the following equation:

$$\Delta F = \sqrt{\left(\frac{Ebh^3}{4L^3}\Delta y\right)^2 + \left(\frac{bh^3 y}{4L^3}\Delta E\right)^2 + \left(\frac{Eh^3}{4L^3}\Delta b\right)^2 + \left(\frac{3Ebh^2}{4L^3}\Delta h\right)^2 + \left(\frac{-3Ebh^3}{4L^4}\Delta L\right)^2} \tag{13}$$

## 2.5 Cantilever Results

The force sensing algorithm was tested with an AFM probe tip that was 450 μm long. A known displacement was applied to the beam using a Queensgate nanopositioner. These displacement inputs were used to calibrate the system so that the virtual force returned by the template matching algorithm corresponded to the deflection at the tip of the beam. The left half of Figure 7 shows a position calibration plot for a 10x objective lens.

The 1 σ prediction intervals are also shown on the plot (meaning that there is a 68.26% chance that the next reading from the vision-based force sensor will be within the interval for a given beam displacement). The maximum uncertainty in the position value given by the template matching algorithm is +/-88.7 nm. Once the system is calibrated to measure the tip displacement, (12) can be used to calculate the actual force that is applied to the beam as discussed in the previous section, and (13) can then be used to calculate the uncertainty in this force value. The right half of Figure 7 shows a force versus displacement plot that was created using the same data that was used to generate the left half plot. Using (13), the maximum 1 σ prediction value for the force measurements is +/- 8.87 nN. Note that in the calculation of the uncertainty in the force value, the uncertainty in the beam parameters was taken to be zero in order to only show the uncertainty in the force calculation due to the template matching algorithm. The system was tested with both a 10x and a 20x objective lens. Figures 8 show the results for the 20x objective lens. Table 1 summarizes the performance of the force sensor with both the 10x objective lens and the 20x objective lens.

**Table 1. Summary of results for visual force sensor.**

| Magnification | NA | Rayleigh Limit (microns) | Pixel Size (microns) | Deflection Prediction Uncertainty (nm) | Force Prediction Uncertainty (nNewtons) |
|---|---|---|---|---|---|
| 10x | .28 | 1.0 | 1.05 | 1σ 88.7<br>2σ 180.5<br>3σ 276.5 | 8.87<br>18.05<br>27.65 |
| 20x | .42 | 0.7 | 0.53 | 1σ 28.5<br>2σ 58.0<br>3σ 88.7 | 2.85<br>5.80<br>8.87 |

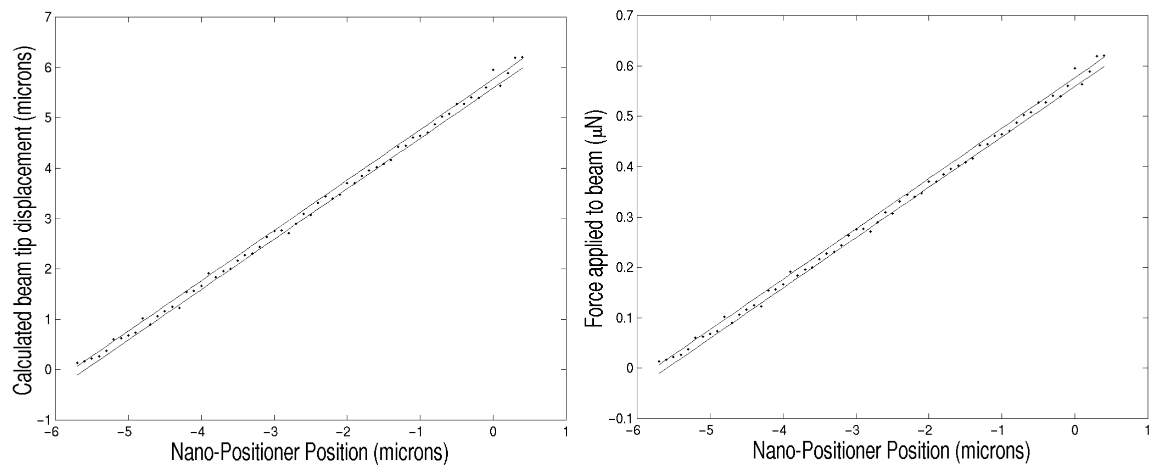

**Figure 7. Calibration plots for cantilever force sensor with 10x lens.**

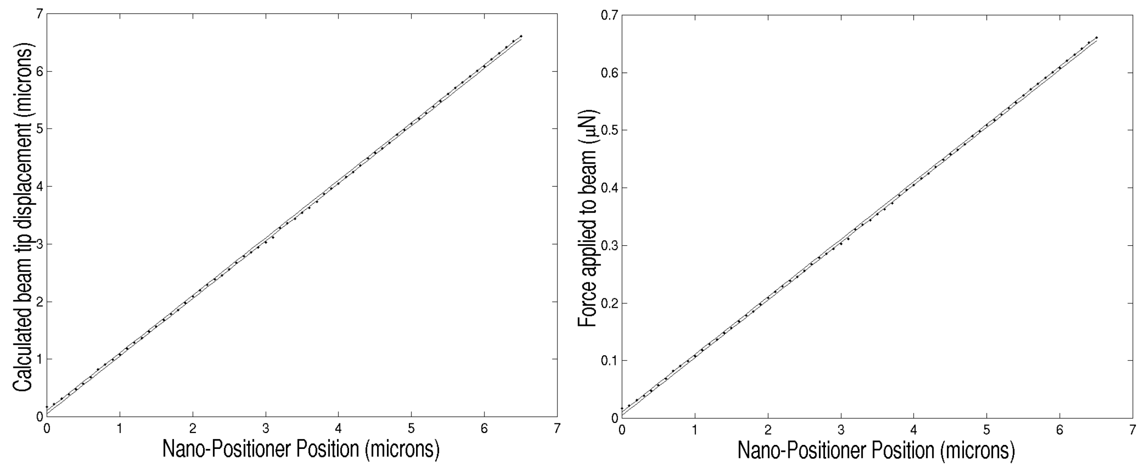

**Figure 8. Calibration plots for cantilever force sensor with 20x lens.**

CHAPTER 3

# Extending VBFM to a Microgripper Device

The methods presented in this thesis can be applied to more complex objects. In order to use this algorithm it is necessary to be able to compute the displacement field of the object as a function of force input. This displacement field might be obtained in a closed form solution, such as was done for the cantilever in (6), or through numerical methods such as FEM. Once this displacement field is known, it can be applied to the template in a way analogous way what was done in (7) through (9) where each template point is displaced as a function of its position in the elastic body and the force applied to the body.

Figure 9 shows a micro-tweezer developed by Yang et al. [27] which is used for a specific microassembly task. The tweezer is cut by micro-wire EDM from spring steel 254 μm thick. A tube is slid over the tweezer to force the jaws of the tweezer together in order to grasp an object. When this tweezer is used for an assembly task it is important to measure the gripping force that the tweezer applies to the object being held.

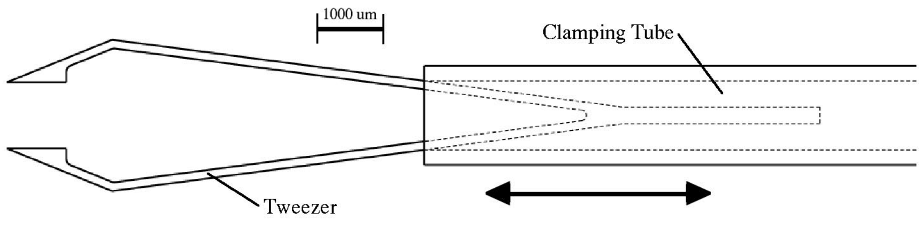

**Figure 9. Micro-tweezer manipulator.**

## 3.1 Modeling of the Micro-Tweezer

Each jaw of the tweezer can be modeled as a cantilever beam. The deflection for each jaw of the tweezer is symmetrical so only one jaw is shown in the deflection model in Figure 10, where $D$ is the displacement of the jaw due to the clamping tube making contact with the tweezer, $F$ is the force applied to the object being held and $\delta$ is the displacement of the jaw at a position $x$ along its length.  is a function of the tube position, $L_2$, and can be written as:

$$D = L_2 \sin\beta - r\cos\beta \tag{14}$$

where $r$ is the inner radius of the clamping tube. Because the tube forces the tweezer jaw to deflect their will be a reaction force $R$ applied to the tube by the jaw. We need to solve for the displacement field, $\delta$, as a function of $x$, $F$, and $L_2$ in order to be able to create the deformable tweezer template. In order to solve for $\delta$ it is first necessary to solve for the reaction force of the tweezer jaw on the clamping tube, $R$. The displacement  applied to

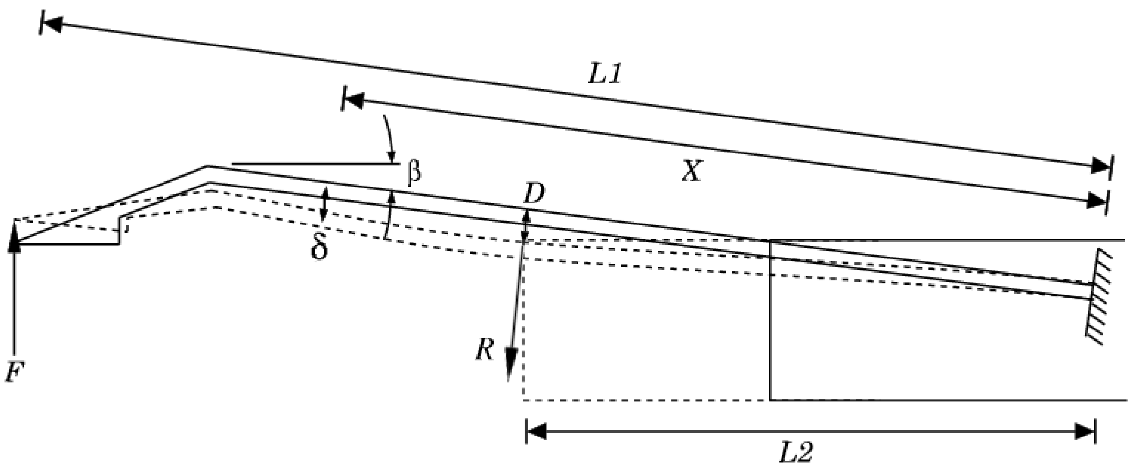

**Figure 10. Deflection model for tweezer jaw.**

the tweezer is a redundant constraint on the cantilever beam so the reaction force $R$ can be solved for by using the beam equation with the constraint that the displacement must be at the location where the tube comes into contact with the tweezer jaw. The reaction force is then found to be:

$$R = \frac{3}{2}\frac{F\cos(\beta)^2 L_1}{L_2} - \frac{1}{2}F\cos(\beta) + 3\frac{EI\cos(\beta)^3\sin(\beta)}{L_2^2} - 3\frac{EI\cos(\beta)^4 r}{L_2^3} \tag{15}$$

Once $R$ is known it is straight forward to calculate the displacement field, $\delta$, of the jaw using the Bernoulli-Euler law. The equations for $\delta$ are shown below. The form of $\delta$ depends what the value of $x$ is relative to the position of the reaction force $R$ and the gripping force $F$.

$$\delta_1 = \frac{1}{6}\frac{F\cos\beta x^2(3L_1 - x)}{EI}d - \frac{1}{6}\left[\frac{1}{2L_2^3}\left(F\cos(\beta)^4 L_1^3\left(2 - 3\frac{L_1 - \frac{L_2}{\cos\beta}}{L_1} + \frac{\left(L_1 - \frac{L_2}{\cos\beta}\right)^3}{L_1^3}\right)\right) + 3\frac{EI\cos(\beta)^3(L_2\sin\beta - r\cos\beta)}{L_3^3}\right]x^2\frac{\left(3\frac{L_2}{\cos\beta} - x\right)}{EI} \tag{16}$$

$$\delta_2 = \frac{1}{6}\frac{F\cos\beta x^2(3L_1 - x)}{EI} - \frac{1}{6}\left[\frac{1}{2L_2^3}\left(F\cos(\beta)^4 L_1^3\left(2 - 3\frac{L_1 - \frac{L_2}{\cos\beta}}{L_1} + \frac{\left(L_1 - \frac{L_2}{\cos\beta}\right)^3}{L_1^3}\right)\right) + 3\frac{EI\cos(\beta)^3(L_2\sin\beta - r\cos\beta)}{L_2^3}\right]L_2^2\frac{\left(3x - \frac{L_2}{\cos\beta}\right)}{\cos(\beta)^2 EI} \tag{17}$$

$$\delta_3 = \frac{1}{6}\frac{F\cos\beta L_1^2(3x-L_1)}{EI} - \frac{1}{6}\left[\frac{1}{2L_3^3}\left(F\cos(\beta)^4 L_1^3\left(2-3\frac{L_1-\frac{L_2}{\cos\beta}}{L_1}+\frac{\left(L_1-\frac{L_2}{\cos\beta}\right)^3}{L_1^3}\right)\right)+3\frac{EI\cos(\beta)^3(L_2\sin\beta - r\cos\beta)}{L_2^3}\right]L_3^2\frac{\left(3x-\frac{L_2}{\cos\beta}\right)}{\cos(\beta)^2 EI} \tag{18}$$

where $\delta_1$ is valid for $0 < x \le L_2/(\cos\beta)$, $\delta_2$ is valid for $L_2/(\cos\beta) < x \le L_1$ and $\delta_3$ is valid for $L_1 < x$. The same set of equations can be written for the bottom half of the tweezer where the displacements will be in the opposite direction. In the above equations, the only two variables that do not depend on the geometry or the material properties of the tweezer are the gripping force $F$ and the position of the tube $L_2$. In order for the vision algorithm to solve for the clamping force, it must solve for the clamping tube position also, therefore (10) becomes:

$$E(S, \theta, T_X, T_Y, F, L_2) = \sum_{j=1}^{N} [(x_{Tj}' - x_{Ij})^2 + (y_{Tj}' - y_{Ij})^2] \tag{19}$$

When this error function is minimized the force applied to the tweezer $F$ and the position of the clamping tube $L_2$ are found.

## 3.2 Micro-Tweezer Results

The tweezer template was first tested on images created by ANSYS where the gripping force was known. One such image is shown in Figure 11. The tweezer was modelled in ANSYS as a two dimensional object in plane stress with constant thickness. The clamp-

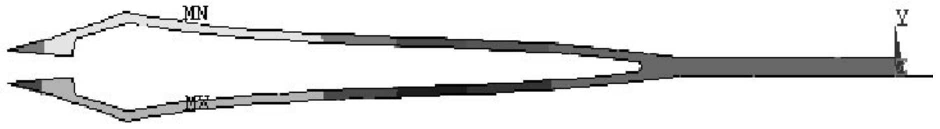

**Figure 11. ANSYS model of tweezer showing deflections due to a specified clamping tube position and part thickness.**

ing tube was modelled by applying a displacement boundary condition to the tweezer jaw of a magnitude given by (14). Instead of applying a force to the tweezer jaw where the object was being held, a part thickness was assumed that provided another displacement boundary condition where the object was being held. When this finite element model is solved, the reaction forces are given as a result. The reaction force where the object is being held is then taken as the gripping force. This process was repeated for a series of increasing clamping tube positions $L_2$ with the same part thickness (much like the situation where a part is being gripped by gradually sliding the clamping tube in order to increase the gripping force). When this series of images created by ANSYS is used as input to the vision algorithm, the vision algorithm should return a gripping force very close to the one that was calculated in the finite element model. Figure 12 shows the gripping force versus clamping tube position from both the ANSYS model and VBFM applied to the output of the ANSYS model.

Equations (16) - (18) can also be verified using the ANSYS model by solving for the clamping force $F$ as a function of the part thickness and the clamping tube position (specifying a part thickness is in effect specifying the value of $\delta$ at the point $x = L_1$, once these values are substituted into (16) - (18), $F$ can easily be solved for in terms of the re-

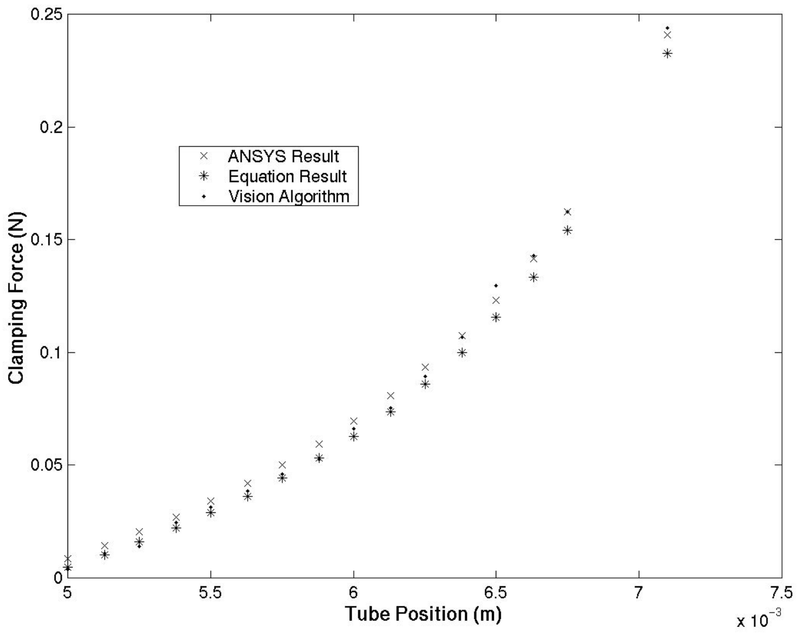

**Figure 12. Results from ANSYS simulation, beam equation and vision algorithm.**

maining variables). Figure 12 also shows the force value predicted by the Bernoulli-Euler law, represented by equations (16) - (18). It can seen from this figure that ANSYS, the beam equations and the vision algorithm (which uses the beam equations for its deflection model) correspond closely.

Next, the vision algorithm was tested on the actual tweezer. A piezoresistive force sensor manufactured by SensorOne (model AE801) was used to verify the output of the vision-based force sensing algorithm. Figure 13 shows the setup that was used. One jaw of the tweezer was applied to the piezoresistive force sensor and the other was applied to a rigid screw. The output of the vision algorithm, along with the output of the piezoresistive force sensor, is shown in Figure 14 versus clamping tube position.

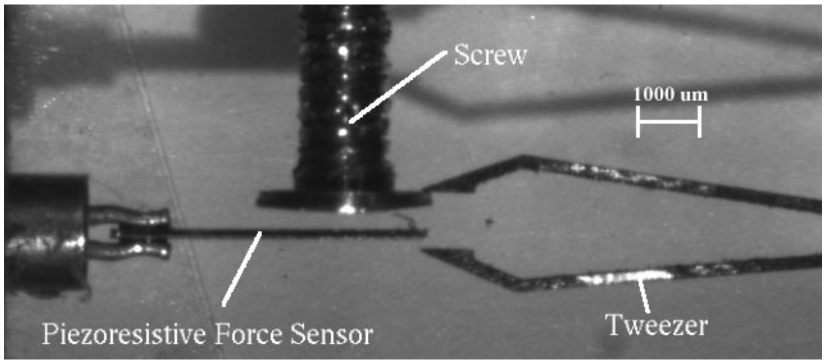

**Figure 13. Setup for verifying vision-based force sensing algorithm applied to the micro-tweezer.**

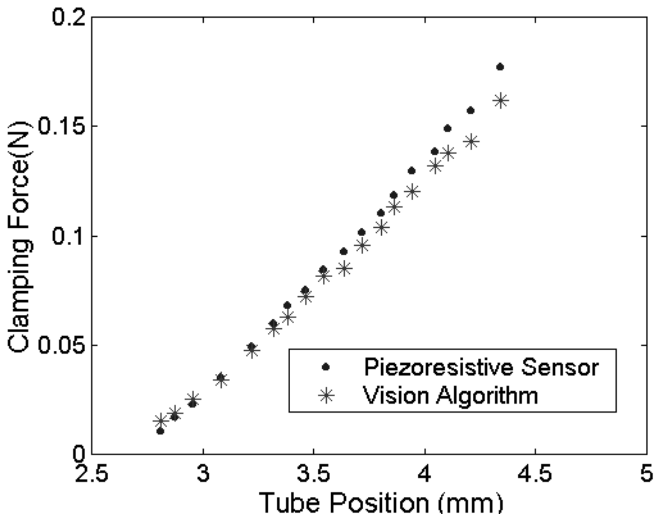

**Figure 14. Vision algorithm output compared to piezoresistive sensor output.**

CHAPTER 4

# VBFM Performance Optimizations

If VBFM is to be used in a control situation it is important that the necessary calculations be performed quickly. The most time consuming step of the template matching algorithm is the evaluation of the error function, (4). It is expensive because for each template vertex, it is necessary to locate the image vertex that is nearest to it. This problem is known as the nearest-neighbor search or the post-office problem. Knowing that the most time consuming step is the evaluation of the error function there are two ways to speed up the algorithm: 1. Reduce the number of evaluations of the error function, or 2. Reduce the time required to evaluate the error function. Both approaches were used to optimize the force sensing algorithm. It is also important to optimize the performance of the Canny edge operator because it is applied to every image that is input into the VBFM algorithm. All of the performance numbers in this section were obtained from an Intel based 800 MHz computer with 128 Mb of RAM.

## 4.1 Canny Edge Operator

As was mentioned previously, the Canny edge operator is used to obtain the binary edge image that is used as the input for the template matching algorithm. The canny edge operator was chosen because it has the ability to create continuous edges. Simple edge algorithms use the derivative of the image in the $x$ and $y$ directions in order to find an edge.

When the magnitude of this derivative is above a certain threshold value an edge is detected. This simple algorithm tends to generate segmented edges. The canny edge operator can detect continuous edges because the operator keeps track of edge directions as well as the image derivative and makes use of two threshold values rather than one [3]. The edge direction information aids the performance of the algorithm because an edge candidate is likely to be an actual edge if it has the same direction as an existing edge. In such a case, the lower threshold value would be used to test whether the pixel is an edge pixel or not. The Canny edge operator has the draw back that it is more computationally expensive then the simple edge detection algorithm.

It is important that the Canny edge operator is calculated quickly in order to optimize the performance of VBFM. In order to obtain the optimum performance the Intel Image Processing Library (IPL) was used in conjuction with Intel Computer Vision Library (OpenCV). The Intel IPL is a collection of low level image routines that make use of the special capabilities that are available with Intel compatible processors. OpenCV provides high level image processing functions (including the Canny edge operator) that make use of the IPL routines. Using these libraries, the Canny edge operator can be performed very quickly. For a full 640 by 480 pixel image the edge operator can be performed in approximately 70 msec (14 Hz) and for a 200 by 200 pixel region the edge operator can be evaluated in approximately 10 msec (100 Hz).

## 4.2 Reducing Number of Error Function Evaluations

The simplest way to speed up the force sensing algorithm is to reduce the number of the times the error function is evaluated. To do this we want a error function minimization algorithm that converges quickly and that evaluates the error function a minimum number of times per iteration.

### 4.2.1 Optimizing the Gradient Search

By using an efficient gradient search algorithm, the iterations needed to find the minimum of the error function can be kept to a minimum. The simplest gradient search method is the gradient decent method. This method has the disadvantage that it does not use the information from previous iterations in choosing the new search direction. As was mentioned previously, the BFGS (Broydon-Fetcher-Goldfarb-Shanno) method is used which does use information from the previous iterations in choosing the new search direction. Both the steepest decent and the BFGS method are known as first-order minimization methods because they use the first derivative of the error function in the attempt to find the minimum of the error function. The traditional template matching algorithm (1) uses a zero-order minimization technique since derivatives of the error function are not used.

All first-order minimization techniques involve a similar overall procedure. An initial estimate $\boldsymbol{x}_0$ at the solution is made, and a search direction $\boldsymbol{S}_0$ is then calculated from the position of $\boldsymbol{x}_0$, where $\boldsymbol{x}_i$ consists of the error function parameters and is written as

$$\boldsymbol{x}_i = \begin{bmatrix} \theta_i \\ S_i \\ T_{X_i} \\ T_{Y_i} \\ F \end{bmatrix} \tag{20}$$

The vector $\boldsymbol{x}_0$ is then incriminated along the search direction $\boldsymbol{S}_0$ according to

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \alpha \boldsymbol{S}_0 \tag{21}$$

where $\alpha$ is a scalar indicating how far to increment $\boldsymbol{x}$ in the search direction. The process for choosing a value for $\alpha$ is a one dimensional optimization problem that can be solved by various one-dimensional optimization techniques which will be discussed in the next section. This process is called a line search and the choice of a line search algorithm is important to the performance of the gradient search. Once $\alpha$ is found, the process repeats itself at the point $\boldsymbol{x}_1$.

First-order minimization techniques differ in how they calculate the search direction $\boldsymbol{S}_i$. The simplest first-order technique is the steepest decent method. In the steepest decent method, the search direction is always set opposite to the gradient of the error function at each iteration. For the steepest decent method, the search direction is given by

$$\boldsymbol{S}_i = -\nabla E(\boldsymbol{x}_i) \tag{22}$$

The steepest decent method is not particularly efficient because it does not use information from the previous iterations in order to find a new search direction. Variable metric methods (such as the BFGS method) use information from the previous iterations in the calculation of each new search direction. Because of this, variable metric methods converge faster than a steepest descent search. The BFGS method uses the following equation to get the search direction at each iteration [25]

$$\boldsymbol{S}_i = -\boldsymbol{H}_i \nabla E(\boldsymbol{x}_i) \tag{23}$$

where $\boldsymbol{H}_i$ is calculated by the following equation

$$\boldsymbol{H}_i = \boldsymbol{H}_{i-1} + \boldsymbol{D}_i \tag{24}$$

$\boldsymbol{D}_i$ is called the symmetric update matrix and it is defined by the following equation

$$\boldsymbol{D}_i = \frac{\sigma + \tau}{\sigma^2}\boldsymbol{p}\boldsymbol{p}^T - \frac{1}{\sigma}[\boldsymbol{H}_{i-1}\boldsymbol{y}\boldsymbol{p}^T + \boldsymbol{p}(\boldsymbol{H}_{i-1}\boldsymbol{y})^T] \tag{25}$$

where $\boldsymbol{p}$, $\boldsymbol{y}$, $\sigma$, and $\tau$ are calculated by the following equations

$$\boldsymbol{p} = \boldsymbol{x}_i - \boldsymbol{x}_{i-1} \tag{26}$$

$$\boldsymbol{y} = \nabla E(\boldsymbol{x}_i) - \nabla E(\boldsymbol{x}_{i-1}) \tag{27}$$

$$\sigma = \boldsymbol{p} \cdot \boldsymbol{y} \tag{28}$$

$$\tau = \boldsymbol{y}^T \boldsymbol{H}_{i-1} \boldsymbol{y} \tag{29}$$

$\boldsymbol{H}_i$ is an approximation to the inverse Hessian matrix which allows the BFGS method to have convergence properties very similar to second order minimization methods. Methods that approximate the Hessian matrix are called quasi-Newton methods.

### 4.2.2 Optimizing the Line Search

The choice of line search algorithm is very important to the performance of the gradient search method. They are many possible line searches that could be used including Newton’s method and the golden section method. When using one of these methods, one often chooses $\alpha$ so that the value of $x_1$ in (21) is minimized. When using quasi-Newton methods this may not be the best aproach because it may actually slow down the speed of convergence of the minimization algorithm and require an unnecessary number of error function evaluations in order to perform the line search. For this reason, the backtracking algorithm can be used to solve for $\alpha$ [5]. The backtracking line search algorithm initially attempts $\alpha = 1$ and uses $\alpha = 1$ if the error function is decreased sufficiently. If the error function is not decreased sufficiently, $\alpha$ is decreased or "backtracked" until a value of $\alpha$ is found that sufficiently decreases the error function. Quadratic interpolation of the error function along the search vector is used for the first backtrack and cubic interpolation is used for subsequent backtracking. The use of the backtracking line search algorithm helps to dramatically reduce the number of error function evaluations.

## 4.3 Optimizing Error Function Evaluation Cost

The second way to optimize the force sensing algorithm is to optimize the evaluation of the error function itself. As was mentioned previously the nearest-neighbor search performed for each template pixel is where the algorithm spends most of its time. In the simplest solution to the nearest-neighbor search, one simply calculates the distance to every image vertex and selects the image vertex with the shortest distance. This algorithm works but makes no use of the spatial structure that exists within the image pixels. If the image vertex data is organized in a spatial data structure, the nearest image pixel can be found without having to measure the distance to every image pixel. The data structure employed to organize the pixel data is the KD-Tree [20]. In addition to using a spatial data structure the error function can be evaluated more efficiently if the number of template pixels are reduced or if the error function is evaluated using more than a single computer processor.

### 4.3.1 KD-Tree Data Structure

A two dimensional KD-Tree is a binary tree data structure where each node corresponds to a data point. Each new node partitions the data space by either a vertical line or a horizontal line. Figure 15 shows a set of points that are partitioned into a KD-Tree. A KD-Tree can be used to find the nearest image vertex with O(log N) operations as opposed to O(N) operations needed to find the nearest pixel without using a spatial data structure (where N is the number of vertex points in the image). The order in which data points are added to the tree is important for achieving the O(log N) performance. In order to achieve the O(log N) performance it is necessary that the KD-Tree is balanced. However, there is a computational cost to building a balanced tree that must be considered when deciding the best method to construct the KD-Tree. The time in creating the KD-Tree becomes important because for every new image frame that is read in from the camera, a new KD-Tree needs to be constructed. The order the data points are added to the KD-Tree determines how balanced the final KD-Tree will be. Three approaches were attempted when choosing the order to add points to the KD-Tree: the no-preprocessing approach where the data points are added in the same order that they were received from the camera which is the scanning order of the original image (an example of such a tree is shown in Figure 16), the balanced approach where the data points are ordered so that a balanced tree is created (see Figure 15), and the randomized approach where the points are placed in random order before they are added to the tree.

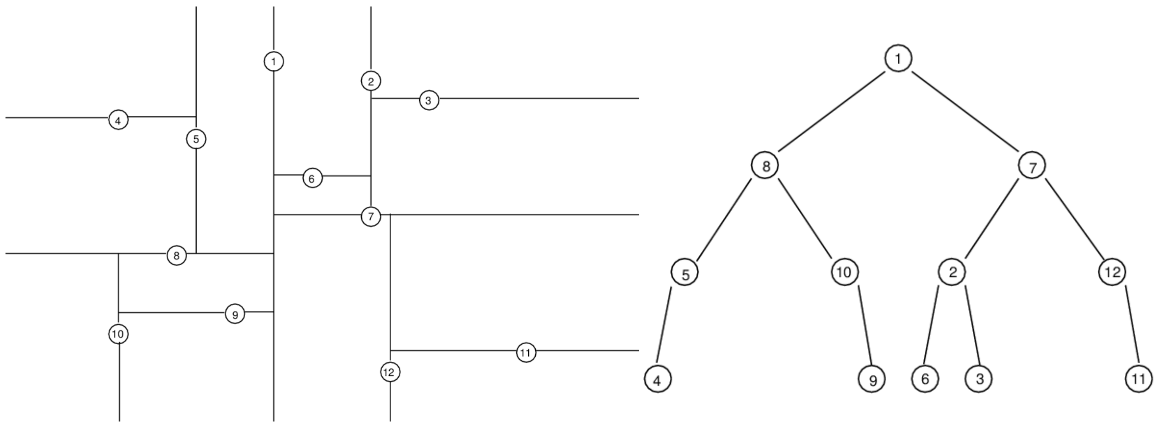

**Figure 15. Partitioned data on left and KD-Tree on right for data that was added to tree in optimal order to create a balanced tree.**

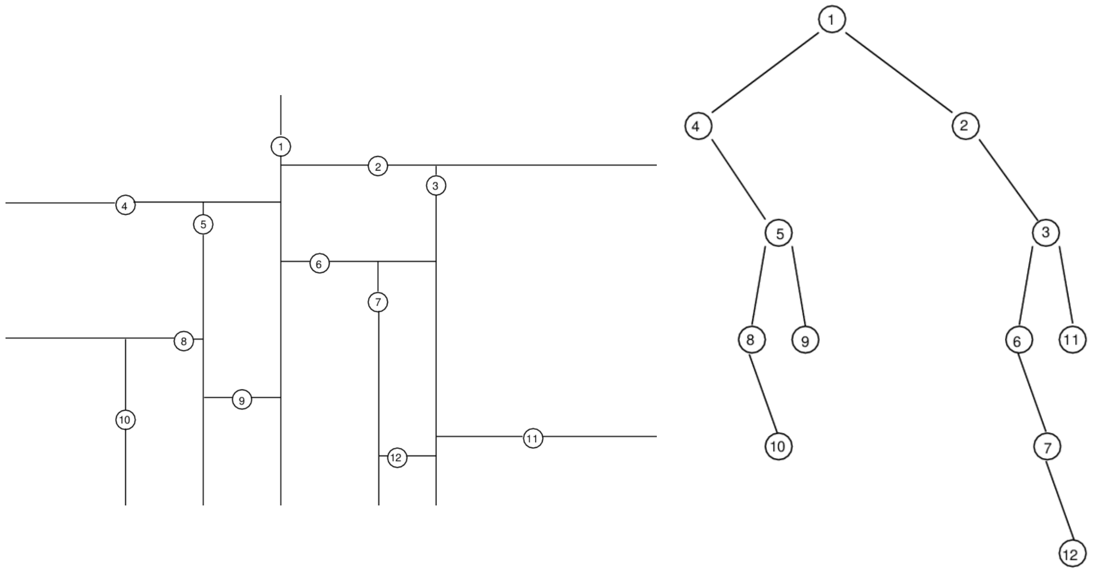

**Figure 16. Partitioned data on left and KD-Tree on right for data that was added to tree in scan line order.**

The results for the three methods of adding nodes to the KD-Tree are summarized in Table 2. The table shows the total time for tree creation and error function minimization time. We want to use the algorithm with the lowest total time. It can be seen from this data that the balanced tree is the most efficient as far as the nearest neighbor searches go since it has the fastest minimization time, however, it takes the longest time to create. The tree with no preprocessing can be created very quickly but it hampers the performance of the minimization. The random insertion order algorithm provides the lowest total time even though it doesn't provide the lowest minimization times. The reason that the no-preprocessing algorithm performs so poorly is because the data is coming from image scan lines so consecutive data points placed into the KD-Tree will have come from nearly the same location on the image (see Figure 16). This will lead to a tree that is very poorly balanced and little advantage will be gained by using the data structure. By randomizing the order of the data entered into the KD-Tree it will be better balanced and the randomized approach also has the benefit that it is not very computationally expensive to randomize the order to the data points before inserting them into the tree.

**Table 2. Comparison of KD-Tree construction algorithms.**

| | Random | Balanced | No Preprocessing |
|---|---|---|---|
| Average Tree Creation Time (sec) | .0082 | .0305 | .0033 |
| Average Minimization Time (sec) | .1217 | .1129 | .1897 |
| Total Time (sec) | .1299 | .1434 | .1930 |

### 4.3.2 Stochastically Under-Sampled Template

The error function can be evaluated more quickly if the number of template pixels are reduced. This reduces the number of nearest neighbor searches that need to be performed. To reduce the number of template pixels, a certain percentage of pixels are removed radomly from the template. Figure 17 shows a cantilever template with all of the edge pixels along with an under-sampled template retaining ten percent of the original edge pixels. Figure 19 shows the average time for force calculation versus the percentage of template pixels that were retained. It can be seen from this plot that there is a linear relationship between the number of template pixels and the speed of the algorithm. A linear relationship is expected because each nearest neighbor search takes approximately the same amount of time, so if there are half the number of nearest neighbor searches the time to perform the minimization should be cut in half (ignoring the time that is required to actually construct the KD-Tree). Also, it can be seen from the plot that the y-intercept of the line is not zero. This is due to the time that is required to construct the KD-Tree before the nearest neighbor search actually begins (the time to perform the Canny edge operator was not included in the results). By reducing the number of template pixels, the time per iteration can be reduced significantly. In this case with 100 percent of the template pixels the average iteration time was approximately 0.12 sec while with 20 percent of the template pixels the average iteration time was reduced to 0.037 sec.

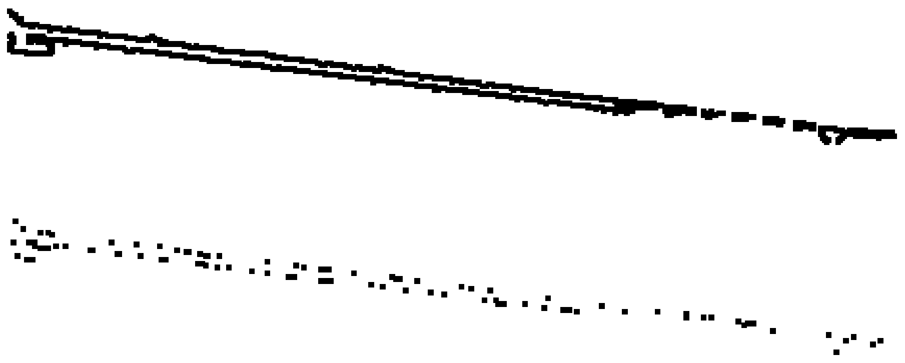

**Figure 17. Full template from edge pixels of image on top and under-sampled template on bottom retaining only 10% of the original edge pixels.**

Reducing the number of template pixels, however, is not without drawbacks. By having less data points in (10), it would be expected that the accuracy of the force result obtained would be decreased. Figure 19 summarizes this by showing the cantilever deflection 1 $\sigma$ prediction intervals versus the percentage of template pixels retained. It can be seen that the prediction intervals become larger when fewer pixels are used in the template. Below 20 percent of pixels, the prediction intervals begin to grow more dramatically. Even when only one percent of the template pixels are retained (such a template would consist of approximately nine pixels because the original template consisted of 920 pixels) sub-pixel deflection prediction intervals of 0.4 μm are still achieved (the pixel size is approximately 1 μm). When determining the percentage of template pixels to use in a particular parameter estimation problem it is important to balance the needs of accuracy and fast calculations.

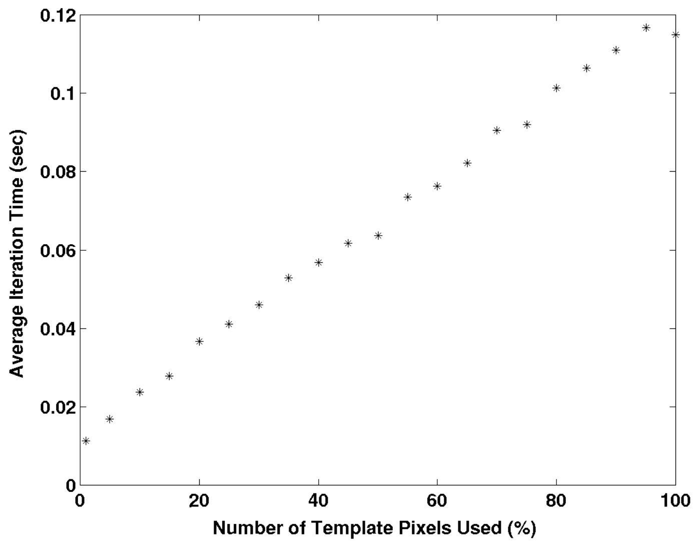

**Figure 18. Under-sampled template performance.**

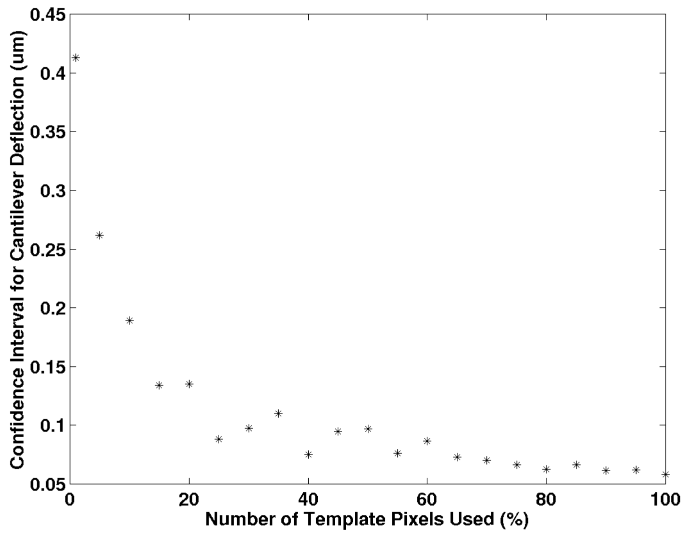

**Figure 19. Under-sampled template effect on system accuracy.**

### 4.3.3 Potential for Parallization Using Multiple Processors

Performance improvements similar to those achieved by decreasing the number of template pixels could be achieved by making use of multiple processors rather than removing template pixels. This could be done by separating the evaluation of the error function (10) into a series of partial sums where the number of partial sums would be the same as the number of processors that are available. Each processor would then evaluate its own partial sum and return the result to the main processor. This method could provide performance benefits similar to those provided by the under-sampled template. For example, if two processors are used the performance should be similar to the performance of the under-sampled template with 50 percent of the original pixels (ignoring the communication cost between processors). With multiple processors, the performance improvements come without the cost of decreased accuracy because all of the template pixels are retained. Even further performance enhancements could be obtained by combining the use of multiple processors with the use of an under-sampled template.

CHAPTER 5

# Robust Estimation Applied to VBFM

In the template matching algorithm discussed above, a least squares error function is used. The choice of error function is important and assumptions are made when a least squares error function is chosen, such as that the errors are normally distributed [7]. However image data errors are often not normally distributed. One common situation where the errors are not normally distributed is when there is occlusion in the image. When there is occlusion, the template pixels that are from the part of the object that is currently occluded will have very large error values compared with those parts of the object that are not occluded. When using a least squares error function, these large errors will dominate the error function even though they may form a minority of the error values. Figure 20 shows the results of tracking a partially occluded object using the least squares measure. Even though only a small part of the object is occluded, the solution found using a least squares error is very far from the true position and orientation of the object. Other situations that may cause a non-normal error distribution include changes in lighting conditions and situations in which the edges from a different object interfere with the edges of the object being tracked.

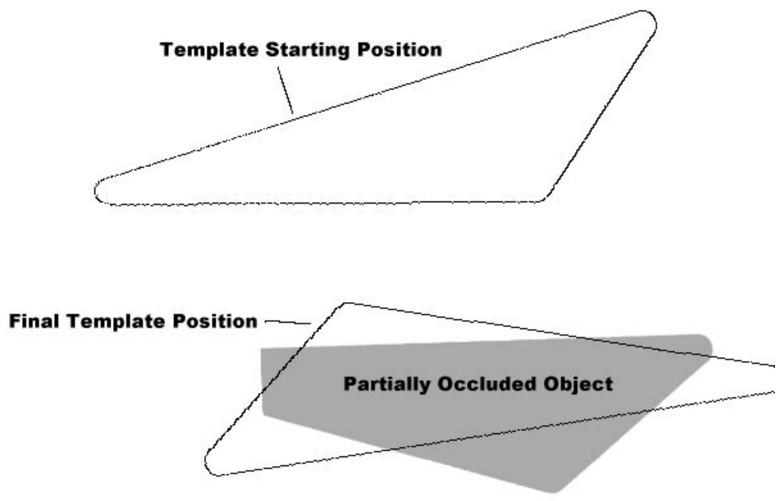

**Figure 20. Tracking occluded object using least squares error measure.**

## 5.1 Robust Estimation

Before discussing the different methods of robust estimation it is necessary to define what is meant by robustness. Huber defines robustness as “insensitivity to small deviations from the assumptions[9]." For example, if there are a small number of outlier points, a robust estimator will still give the correct estimate for the desired parameters. Using this definition, the least squares error measurement is not robust because a single outlier point can cause significant error in the parameter estimate. The least squares error function is:

$$\sum r_i^2 \tag{30}$$

where $r_i$ is the error between the model and the data (oftentimes called the residual). This type of estimation is sometimes referred to as L2 estimation. The problem with the least squares approach is that since the value of the error is squared, any outlier points will dominate the value of the error function. For this reason, a more robust estimator takes the sum of the absolute values of the error rather than the square of the error:

$$\sum |r_i| \tag{31}$$

This estimation technique is referred to as L1 estimation. The L1 estimator is more robust to outliers that the L2 estimator. It should be noted, however, that since the absolute value function does not have a continuous derivative, this type of estimator cannot be used with a gradient based minimization method such as the BFGS method.

Another group of robust estimators that are commonly used are called M-estimators. They are called M-estimators because they are based on the maximum likelihood estimate. An M-estimator takes the form:

$$\sum \rho(r_i) \tag{32}$$

where $\rho(r)$ is a symmetric function with a global minimum at zero [19]. If the distribution of the error is know in advance, the function $\rho(r)$ can be determined in advanced using maximum likelihood techniques. The L2 estimator is actually arrived at after applying maximum likelihood to a Gaussian error distribution [7]. Unfortunately the exact error distribution is not usually known in advance. In that case, a $\rho(r)$ is chosen that has robust characteristics.

The least median of squares (LMS) estimator is another frequently used robust estimator. The estimator can be written as:

$$median(r_i^2) \tag{33}$$

By taking the median rather than the sum of all of the errors, the estimator becomes very robust. In fact, it can be shown that the breakdown point of the LMS estimator is 50% [19] The breakdown point is defined as the smallest fraction of outliers that are required to cause the estimated parameters to be arbitrarily far from their true values. The maximum possible breakdown point achievable by any estimator is 50%. The breakdown point for all M-estimators is 0%, meaning that a single outlier point that is far enough away can move the estimates arbitrarily far from their true values. However, some M-estimators are more robust to outliers than others depending on the distribution it is based on. The LMS error function is not differentiable so the minimization of the error function is computationally expensive. Statistically based minimization techniques are usually used to solve LMS problems.

## 5.2 Robust Estimation in Computer Vision

Various robust techniques have been applied in computer vision. The Hough transform [12], a common method used to detect straight lines where a voting technique is used to determine the most likely value for the two unknown parameters describing the line, is an example of a robust technique. It turns out that by choosing the parameter values with the most votes, the Hough transform behaves much like an M-estimator [22]. Robust techniques have also been used to estimate object surfaces from a range image [22]. A range image contains distance values rather than illumination values for each of its pixels. Range data can be corrupted by specular highlights or by thresholding effects thus requiring the use of robust estimation techniques.

Robust techniques have also been used for pattern matching within images. Hager et. al. [8] demonstrated a method to track objects in two dimensions using robust techniques that was robust to illumination changes and partial occlusion of the object. This method, however, could not be used to measure forces because it makes use of a template that stores illumination values rather than edges. It is not possible to apply a displacement model such as the cantilever model to an illumination image as in the manner performed with the edge template. This method also has the limitation that it is assumed that the object being tracked moves a small distance between each image frame. Lai [11] demonstrated a method to determine the affine transformation of an object in three dimensions using robust statistical techniques that was robust to illumination changes and to partial occlusion of the object. This method also makes use of a template that stores illumination values so it could not be applied to VBFM.

## 5.3 Robust Estimation Applied to Template Matching

Even though the LMS estimator is the highest possible breakdown point, it cannot be used for this force sensing application because it is computationally expensive. An M-estimator can be used, though, if the function $\rho(r)$ is differentiable. One M-estimator that is differentiable and has desirable robust characteristics is based on the Cauchy distribution. For the Cauchy distribution, the function $\rho(r)$ is [22]:

$$\rho(r) = \log\left(1 + \frac{1}{2}r^2\right) \tag{34}$$

The Cauchy distribution has the property that it is a heavily tailed distribution allowing for outlier points. The difference between the Gaussian and Cauchy distribution can be seen in Figure 21.

## 5.4 Results of Using Robust Estimation

When the Cauchy robust estimator is used with the image in Figure 20 the correct location of the occluded object is found as shown in Figure 22. These two tests were performed with the same initial template location. The robust estimator was also tested when measuring the deflection of the cantilever with the presence of occlusion. The template was defined when there was no occlusion present and then part of the cantilever was occluded before its deflection was measured by the vision algorithm. This was done with both the

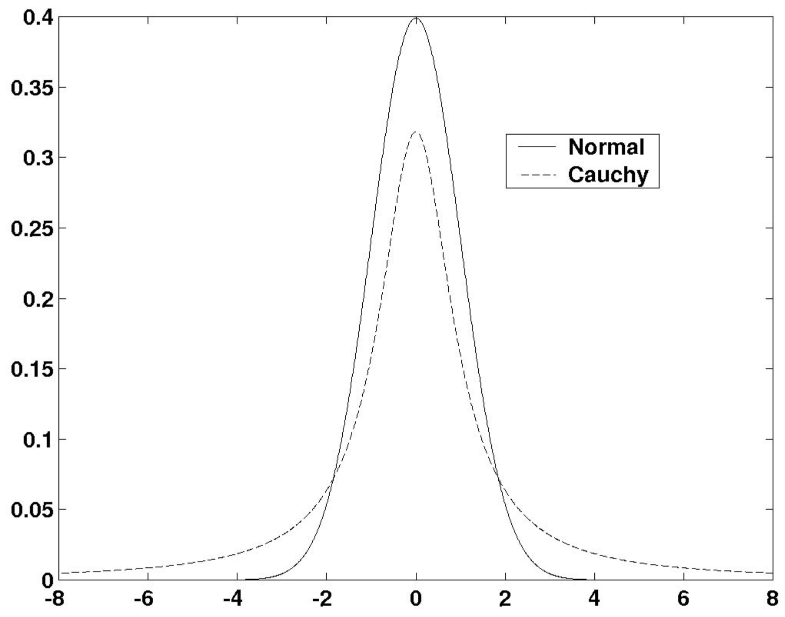

**Figure 21. Probability density functions for the Cauchy distribution and the Normal distribution.**

least squares estimator and the Cauchy estimator. The portions that were occluded are shown in Figure 23. Occluded Region 1 represents 10 percent cantilever occlusion and Occluded Region 2 represents 30 percent cantilever occlusion. The deflection calibration plots for both of the estimators are shown in Figure 24 for Occluded Region 1 and in Figure 25 for Occluded Region 2. Table 3 summarizes the results for both of the estimators. The performance of the robust estimator remains approximately the same whether occlusion is present or not. The performance of the least squares estimator deteriorates when 10 percent occlusion is present and the least squares estimator fails entirely for the 30 percent occlusion case (as can be seen in the left half of Figure 25). It is also interesting to note that even when there is no occlusion, the robust estimator performs better than the least squares estimator. This indicates that even when there is not occlusion the errors in the image were not normally distributed. If the errors had been normally distributed the least squares would have performed better because it is the maximum likelihood estimator for the normal distribution.

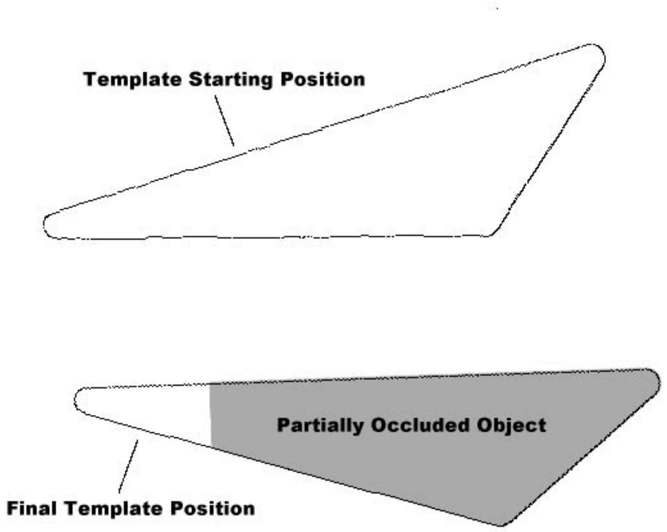

**Figure 22. Tracking occluded object using robust Cauchy M-estimator.**

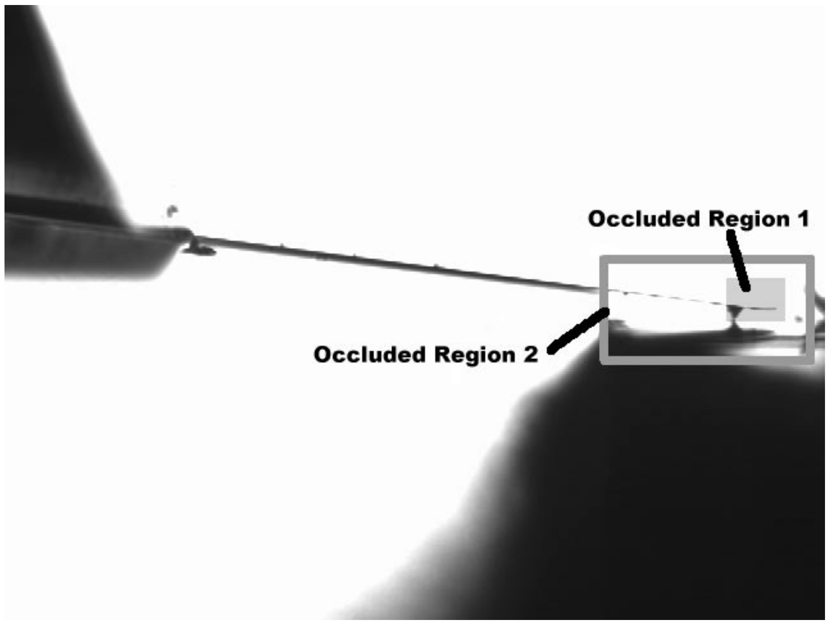

**Figure 23. Regions of image that were occluded to test robust estimator.**

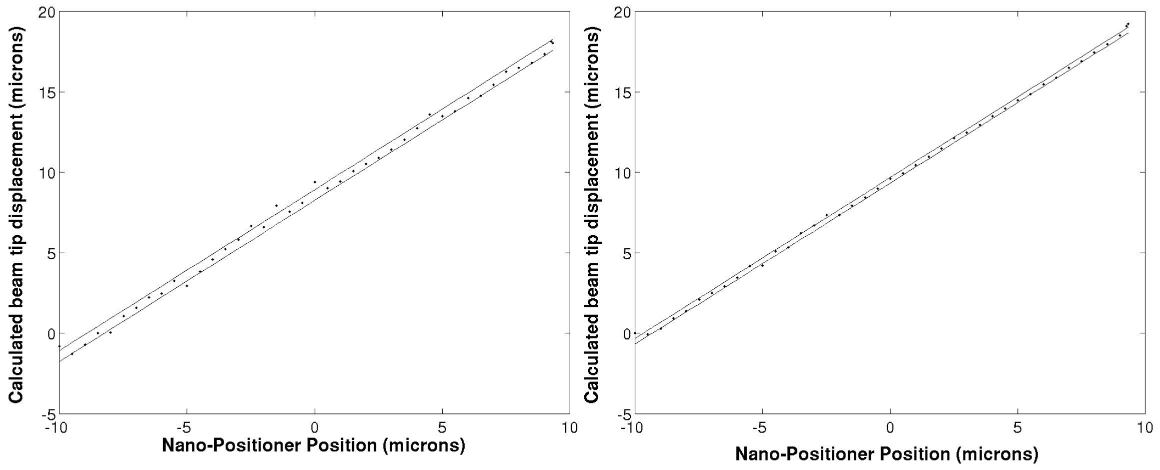

**Figure 24. Displacement calibration plots for the 10% occluded input images. Least squares estimator on left and Cauchy estimator on right.**

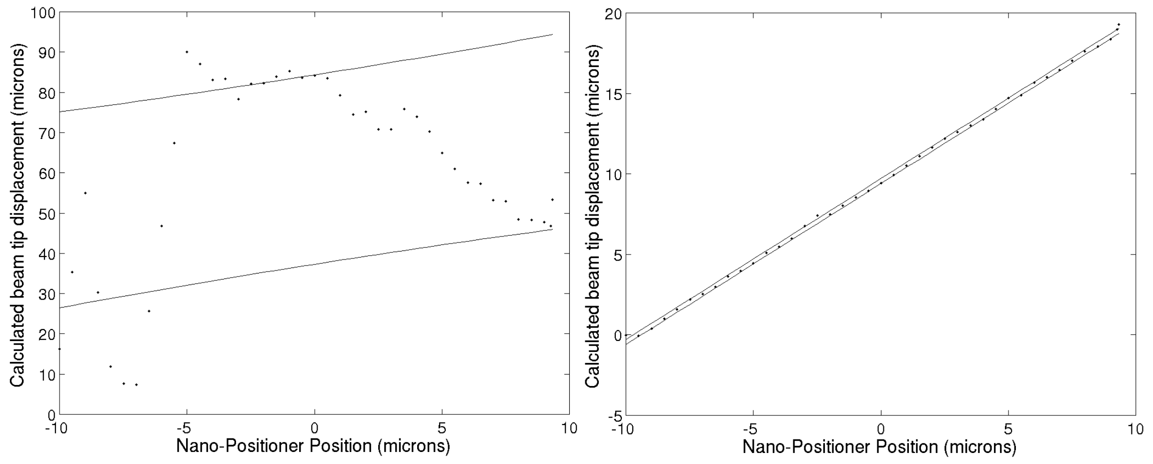

**Figure 25. Displacement calibration plots for the 30% occluded input images. Least squares estimator on left and Cauchy estimator on right.**

**Table 3. Summary of results for occlusion test of robust estimator.**

| | Least Squares Deflection Uncertainty (microns) | Cauchy Deflection Uncertainty (microns) |
|---|---|---|
| With no Occlusion | 1σ 0.2108<br>2σ 0.4320<br>3σ 0.6699 | 0.1790<br>0.3668<br>0.5688 |
| With Occluded Region 1 | 1σ 0.3432<br>2σ 0.7033<br>3σ 1.0905 | 0.1775<br>0.3637<br>0.5640 |
| With Occluded Region 2 | 1σ 24.309<br>2σ 49.811<br>3σ 77.235 | 0.1607<br>0.3294<br>0.5107 |

CHAPTER 6

# VBFM Using Neural Networks

In the case of the cantilever and the tweezer, a closed form solution for the displacement field was easily obtained. Often times it is not possible to easily obtain a closed form solution. This is true for objects with complex geometries, objects with non-linear stress-strain relationships, objects with large deflections (finite strains) and when the material properties are not known. For such objects it would be desirable to learn the displacement field of the object by simply observing how the object behaves under a known force input. Once the properties of the object are learned then the object could be used as a force sensor. Figure 26 shows a schematic of this learning aproach. Feed-forward neural networks with two layers provide the ability to accomplish this task because they are able to approximate any continuous functional mapping relationship to arbitrary accuracy, given that there are a sufficient number of hidden nodes in the network [1]. In this chapter it is demonstrated how neural networks can be used to enable VBFM to be used with objects where the closed form solution of the displacement field can not be easily obtained.

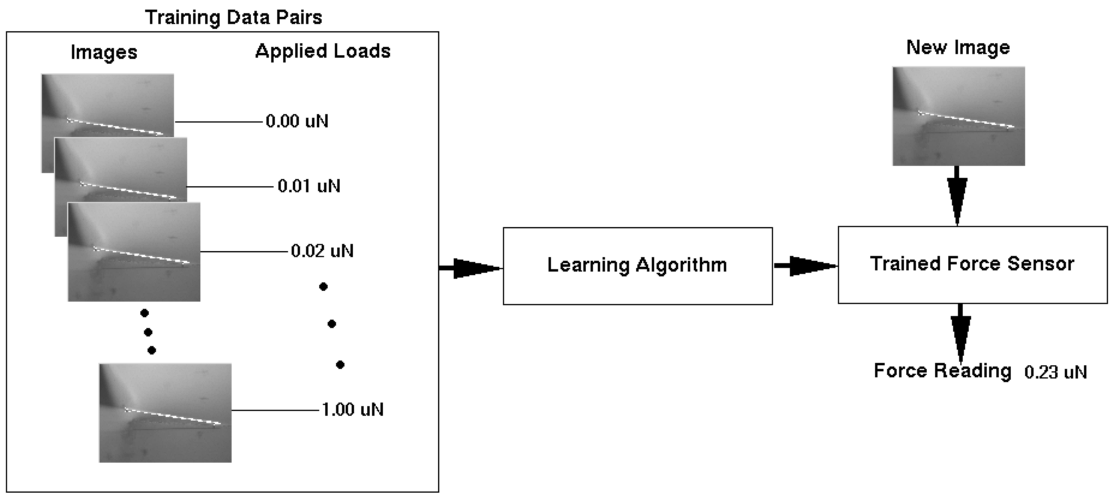

**Figure 26. Learn by seeing approach to visual force sensing.**

The neural networks used are feed-forward two layer neural networks with the layout shown in Figure 27. They were trained using the error back-propagation method in conjunction with the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method to find each new search direction. Each hidden node has a logistic sigmoid activation function of the form

$$g(a) = \frac{1}{1 + \exp(-a)} \tag{35}$$

where $a$ is sum of all of the weighted inputs to the node. The output nodes have a linear activation function which simply returns the sum of all of the node's weighted inputs.

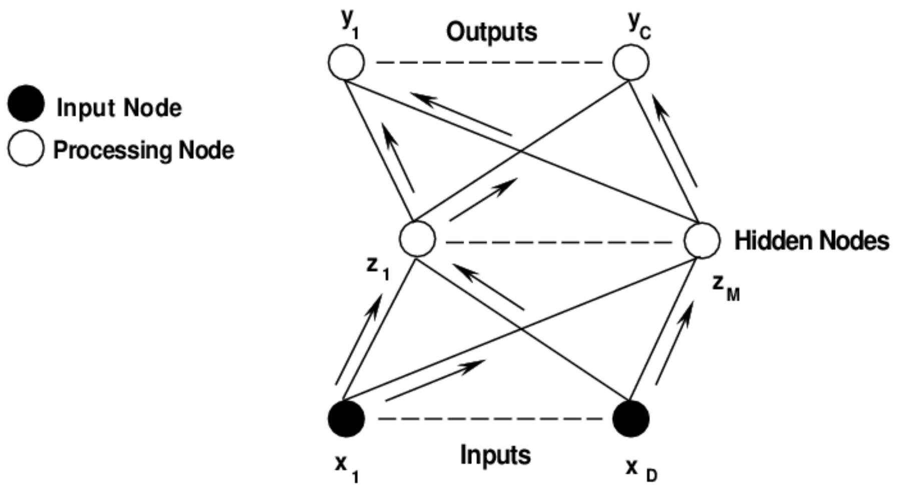

**Figure 27. Neural network diagram.**

## 6.1 Sending Pixel Data Directly to Neural Network

The most straight-forward approach to vision-based force sensing with neural networks is to use image pixels as the input of the neural network and to make force the output of the neural network. This approach has the drawback that there are a very large number of inputs to the network (307200 inputs for a 640 by 480 pixel image). Having many inputs makes the network inefficient, and it will be unlikely that the network will be trainable by gradient optimization methods because of the large number of local minima that will likely occur in the search space. It is desirable to pre-process the pixel data so that fewer variables can capture the information that is contained with the image. The aproach that is taken to pre-process the image data is to take the discrete cosine transform (DCT) of the image. This aproach was also used by Pan et al. [17] to pre-process pixel data for the application of face recognition. The DCT coefficients $C[i, j]$ can be found using:

$$C[i, j] = \sqrt{\frac{4}{MN}} \sum_{x}^{M} \sum_{y}^{N} I[i, j] \cos\frac{(2y+1)i\pi}{2M} \cos\frac{(2x+1)j\pi}{2N} \tag{36}$$

where $I[i, j]$ is the pixel value at the coordinate $(i, j)$, $M$ is the width of the image and $N$ is the height of the image. As can be seen from (36), the DCT of an image has the same number of coefficients as the original image had pixels. If the inverse DCT is applied to the coefficients, the original image will be returned. The DCT components that represent the high frequency content of the image tend to be small in value and can often be ignored. The image that is reconstructed will still contain most of the information that the original image contained. This is the process that is used to compress images for the JPEG image format [15]. This same process can be used to reduce the number of inputs that are needed for the neural network.

A small number of DCT components contain most of the information in the image. It is necessary to have a systematic method to choose which DCT components to pass on to the neural network. To find the DCT components that were most influenced by the force applied to the object, the statistical correlation was computed between each of the DCT coefficients and the force applied to the beam for a set of input-output training pairs. It was discovered that the DCT coefficients had a stronger correlation with the force value if the Canny edge image was used as the input to the DCT rather than the raw pixel values. Figure

28 shows the flow diagram for this neural network approach (in this figure bold arrows indicate vector values).

### 6.1.1 Results for DCT Aproach

This approach was applied to the same cantilever beam that was used in Chapter 2. Known displacements where applied to the beam (thus the force applied to the beam was known since the stiffness of the beam was known). The number of training pairs that were used is 131 with randomly selected force values between 0 and 1.3 uN. Fifteen of the DCT components were used and the network had 10 hidden nodes. After training the network with the 131 training pairs, the network was shown 66 additional images of the beam with random forces applied between 0 and 1.3 uN (note that these images where not used in training of the network). The position calibration plot is shown in left half of Figure 29 and the force calibration plot is shown in right half of this figure. Table 4 summarizes the results with this network approach. The accuracy of the neural network approach is very similar to the template approach used in Chapter 2.

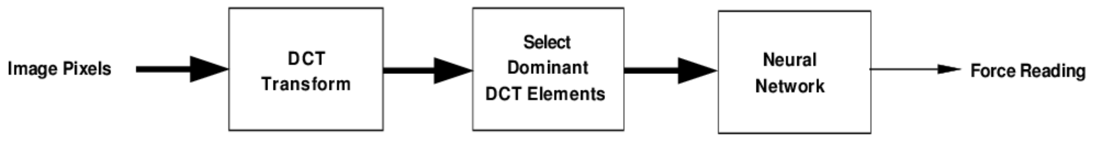

**Figure 28. Flow diagram for DCT neural network approach.**

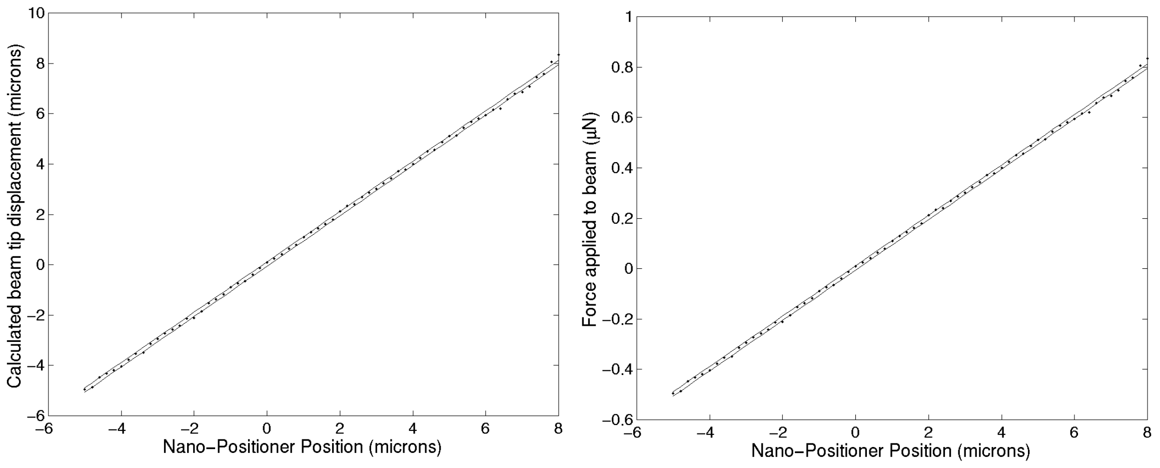

**Figure 29. Position and force calibration plots for DCT neural network approach.**

**Table 4. Summary of results for neural network force sensor using DCT approach.**

| Magnification | NA | Rayleigh Limit (microns) | Pixel Size (microns) | Deflection Prediction Uncertainty (microns) | Force Prediction Uncertainty (μNewtons) |
|---|---|---|---|---|---|
| 10x | .28 | 1.0 | 1.05 | 1σ 0.0886<br>2σ 0.1801<br>3σ 0.2756 | 0.0089<br>0.0180<br>0.0276 |

### 6.1.2 Drawbacks for the DCT Aproach

Even though it was shown that the neural network could successfully learn the deflection properties of the cantilever beam there are some drawbacks of sending DCT data to the neural network. One drawback is that unlike the template matching aproach, this neural network approach is not invariant to object translation or rotation. This method is also very sensitive to lighting conditions. These drawbacks occur because the lighting conditions, translations and rotations affect the DCT coefficients in unknown ways that cannot be easily subtracted. Because of this it is desirable to find a neural network approach that is invariant to such aesthetic changes in the input image.

## 6.2 Using Neural Networks Within the Template Matching Framework

It would desirable to use the learning techniques discussed above in conjuction with the existing template matching procedure. Previously, as in the cantilever application, the model for the elastic body was applied directly to the template pixels as was shown in (7) - (9). If a neural network could be trained with the displacement field for an elastic object, the neural network could be used to deflect the template pixels before the homogeneous transform $A$ is applied to the template. Such a neural network would take the template pixels $x$ and $y$ coordinates $(x_T, y_T)$ and applied load $F$ as inputs and would output the displaced template coordinates $\overline{x_T}$ and $\overline{y_T}$ as shown by the flow diagram in Figure 30. The homogeneous transform $A$ could then be applied to these displaced pixel coordinates as follows

$$\begin{bmatrix} x_T' \\ y_T' \\ 1 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} \overline{x_T} \\ \overline{y_T} \\ 1 \end{bmatrix} \tag{37}$$

The transformed template pixels can now be used with the error function in (10) in the same matter that was done with the cantilever and microtweezer examples previously. The neural

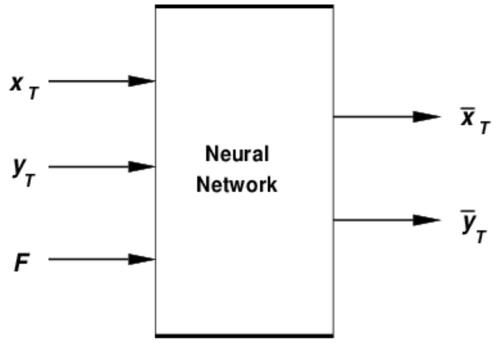

**Figure 30. Flow diagram for neural network used within template matching framework.**

network could be trained with image data by showing how the pixels of the object move as a force is applied to the object. This would require tracking features on the object to see how they deform. This type of neural network could also be trained using an elastic model such as the beam equation (6) or it could even be trained using a finite element simulation. So even though a finite element simulation is not fast enough to compute in real time, its information could be stored in a neural network which could be evaluated in real time. The previous approach to using neural networks could not be trained using a finite element model because it had DCT coefficients as inputs and it would be very difficult to reconstruct these DCT coefficients because it would require modelling the camera and microscope optics. This method also retains the benefits of the template matching process including invariance to translation and rotation and handling of occlusion or noise using robust statistics.

The previously described neural network approach was applied to the cantilever beam. The network was trained directly with the beam equation rather than from actual image data. The number of training pairs used was 10,000 and the number of hidden nodes used was 20. The neural network template was compared directly to the template based on the beam equation represented by (7)-(9). The results of this comparison are shown in Table 5 where it can be seen that the neural network template and the beam template have nearly identical performance.

**Table 5. Summary of results using neural network within template matching framework.**

| | Beam Model Template | Neural Network Template |
|---|---|---|
| Deflection Uncertainty (microns) | 1σ 0.2115<br>2σ 0.4333<br>3σ 0.6718 | 0.2094<br>0.4291<br>0.6654 |

CHAPTER 7

# Conclusions

It has been shown that VBFM can be used to accurately measure forces in the micro and nanonewton ranges. Using this vision based approach we have shown that forces applied to an AFM cantilever beam can be measured to within a 1 $\sigma$ uncertainty of +/- 3nN. This level of accuracy can be achieved because the template matching algorithm matches the image to sub-pixel resolution. We also demonstrated the use of this approach with a micro-tweezer to measure forces in the micronewton range. A key aspect of VBFM is how simply it can be applied to many different elastic objects without having to physically change the object. The microtweezer is a good example of this because, even though the tweezer was not originally designed to be used with VBFM, it was successfully used with it.

Two important aspects of any computer vision algorithm that is intended to be used for control purposes is that it can be performed quickly and that it is robust to noise within the image. Various techniques were used within the VBFM framework in order to allow for real-time performance. These techniques include the use of quickly converging error minimization techniques and spatial data structures. Higher speeds can also be achieved by using an under-sampled template, however this method reduces the accuracy of the force measuring algorithm. These techniques in combination give VBFM real-time performance potential. In order for a computer vision algorithm to be useful it must be robust to occlusions that may occur and image noise. The use of robust estimation techniques based on M-estimators help to make VBFM handle occlusion of up to 30 percent. The efficiency of VBFM and its robust properties make the approach useful for control applications.

In order to make VBFM applicable to a wider range of elastic objects, neural networks were used to learn the elastic properties of an object. This can be useful if the elastic properties of an object are not known in advance. It was shown how neural networks can be used within a template matching framework so that all benefits of template matching can be retained including invariance to affine transforms and robustness to occlusion and image noise. This neural network aproach allows VBFM be used with a much wider range of objects.

VBFM provides a straight forward method for using an elastic object as a force sensor. This capability is important for micromanipulation and microrobotics because force sensing is essential for these tasks. As MEMS devices become more complex, microassembly will play an increasing role, and it will to be essential to have reliable force sensing technology. VBFM provides this technology using equipment that often already exists within a micromanipulation workstation.

# References

[1] Bishop, C. M., *Neural Networks for Pattern Recognition,* Clarendon Press, Oxford, 1995.

[2] Born, M., Wolf, E., *Principles of Optics*, Pergamon Press Ltd., Oxford, 1965.

[3] Canny, J., “A Computational Aproach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, November, 1986.

[4] Danuser, G., Mazza, E., “Observing Deformations of 20 Nanometer With a Low Numerical Aperture Light Microscope,” *Optical Inspection and Micromeasurements*, vol. 2782:180-191 SPIE, 1996.

[5] Dennis, J.E., Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and Non-linear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[6] Dong, L., Arai, F., Fukuda, T., “3D Nanorobotic Manipulations of Multi-Walled Carbon Nanotubes,” *Proc. 2001 IEEE Int. Conf. of Robotics and Automation (ICRA2001)*, Seoul, Korea, May 2001.

[7] Draper, N. R., *Applied Regression Analysis*, 3rd Edition, John Wiley and Sons, Inc., New York, 1998.

[8] Hager, G. D., Belhumeur, P. N., “Efficient Region Tracking With Parametric Models of Geometry and Illumination,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, no. 10, October, 1998.

[9] Huber, P. J., *Robust Statistics,* John Wiley & Sons, New York, 1981.

[10] Kass, M., Witkin, A., Terzopoulos, D., “Snakes: Active Contour Models,” *International Journal of Computer Vision*, pp. 321-331 1998.

[11] Lai, S., “Robust Image Matching under Partial Occlusion and Spatially Varying Illumination Change,” *Computer Vision and Image Understanding,* 78, pp. 84-98, 2000.

[12] Leavers, V.F., “Survey: Which Hough Transform?,” *CVGIP: Image Understanding,* Vol. 58, No. 2, pp. 250-264, September, 1993.

[13] Metaxas, D., *Physics-Based Deformable Models*, Kluwer Academic Publishers, Boston, Mass, 1997.

[14] Meer, P., Mintz, D., Rosenfeld, A., “Robust Regression Methods for Computer Vision: A Review,” *International Journal of Computer Vision,* 6:1, pp. 59-70, 1991.

[15] Miano, J., *Compressed Image File Formats*, Addison-Wesley, Reading, Massachusetts, 1999.

[16] Nelson, B., Zhou, Y., Vikramaditya, B., “Sensor-Based Microassembly of Hybrid MEMS Devices,” *IEEE Control Systems*, December 1998.

[17] Pan, Z., Rust, A. G., Bolouri, H., “Image Redundancy Reduction for Neural Network Classification using Discrete Cosine Transforms,” *Proceedings of the International Joint Conference on Neural Networks*, IEEE, vol. 3, pp. 149-154, 2000.

[18] Pratt, W., *Digital Image Processing*, John Wiley & Sons, New York, 1991.

[19] Rousseeuw, P. J., *Robust Regression and Outliear Detection,* John Wiley and Sons, New York, 1987.

[20] Samet, H., *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, Mass., 1990.

[21] Sokolnikoff, I., *Mathematical Theory of Elasticity*, Krieger Publishing Company, Malabar, Florida, 1983.

[22] Stewart, C. V., "Robust Parameter Estimation in Computer Vision," *SIAM Review,* vol. 41, no. 3, pp. 513-537, 1999.

[23] Tortonese, M., Yamada, H., Barrett, R C., Quate, C F., "Atomic force microscopy using a piezoresistive cantilever," *Transducers 91 International Conference on Solid State Sensors and Actuators*, pp. 448-451 1991.

[24] Tsap, L., Goldgof, D., Sarkar, S., "Efficient Nonlinear Finite Element Modeling of Nonrigid Objects via Optimization of Mesh Models," *Computer Vision and Image Understanding*, Vol. 69, pp. 330-350, March 1998.

[25] Vanderplaats, G., *Numerical Optimization Techniques for Engineering Design*, McGraw-Hill, New York, 1984.

[26] Wang, X., Ananthasuresh, S., and Ostrowski, J. "Vision-Based Extraction of Geometry and Forces From Fabricated Micro Devices," *Technical Proceedings of the MSM 2000 International Conference on Modeling and Simulation of Microsystems,* March 2000.

[27] Yang, G., Gaines, J. A., Nelson, B. J., "A Flexible Experimental Workcell for Efficient and Reliable Wafer-Level 3D Microassembly," *Proc. 2001 IEEE Int. Conf. of Robotics and Automation (ICRA2001)*, Seoul, Korea, May 2001.

[28] Yuille, A., Cohen, D., Hallinan, W., "Feature extraction from faces using deformable templates," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 99-111, 1992.